# Framework for the Performance Assessment of Architectural Options on Intelligent Distributed Applications

Günter Haring‡, Carlos Juiz*, Christian Kurz‡, Ramon Puigjaner*, Joachim Zottl‡
‡ Department for Computer Science and Business Informatics, University of Vienna
Lenaugasse 2/8, A-1080 Vienna, Austria
{guenter.haring, christian.kurz, joachim.zottl}@univie.ac.at
*Department of Mathematics and Computer Sciences, University of Balearic Islands
Carretera de Valldemossa, km. 7.5, 07071 Palma de Mallorca, Spain
{cjuiz, putxi}@uib.es

## ABSTRACT

This position paper brings together the evaluation of ambient intelligence architectures in context-awareness systems with performance modeling. Thus, firstly appropriate description methods for distributed intelligent applications are summarized. Derived from the system characterization, typical software performance engineering techniques are based on the augmented description of the model regarding performance annotations. However, these annotations are only related with the syntactical view of the architecture. In the next generation of performance assessment tools for intelligent context-awareness systems, the description of the system would be capable of reasoning and acquiring knowledge about performance. Having an appropriate architectural description including performance aspects, any possible design options for intelligent distributed applications can be evaluated according to their performance impact. Therefore, we propose the use of an ontology with performance-related information - not only to evaluate the architecture off-line - but also building a context broker that assesses the performance during execution.

**KEYWORDS**: *performance evaluation, distributed software performance engineering, context-awareness, ambient intelligence, mobile devices*

## 1. INTRODUCTION AND MOTIVATION

To be able to create architectures for intelligent distributed systems one has to consider the capabilities and limitations of the devices running the applications. One fundamental aspect is performance issues which have to be included into the decision process when choosing between different architectural options. Performance analysis of architecture options should be integrated in early life cycle stages of a software development process [8].

The term *software architecture* (*SA*) of a program defines the systems structure, which comprises the software components, their external observable behavior and the relationship of these components to each other [1], [3]. A software design method is a systematic approach for creating a system design. During a given design step, the method may provide a set of structuring criteria to help the designer in decomposing the system into its components [4]. However, non functional features of the system, e.g. performance, have not been considered for those software design methodologies. Thus, the performance modeling of systems is based on a certain type of conceptual performance formalism (e.g. queuing networks (QN) and their extension (EQN), stochastic timed Petri nets (SPTN) or stochastic process algebra (SPA)). As the size and complexity dramatically increase, many software (distributed) systems can not provide performance properties as required due to fundamental architecture or design problems. During the last years the UML (Unified Modeling Language) has been widely used to specify, construct and document the functionality of software systems [15]. In order to reduce the gap between functional models and performance evaluation, a software and performance community has emerged to provide (automatically) accessible techniques and tools to include performance annotations for building performance prediction constituting a new topic in Software and Performance Engineering (SPE) [9].

UML diagrams provide key information required for performance analysis so that they describe both behavior and resources. Therefore, sequence, activity, state chart and deployment annotated diagrams may be annotated to express some performance information in a direct or indirect way [19], [20], [21].

In typical software architectures of distributed systems communication between clients and servers has an important role. However, the growing availability of mobile and wireless networks and the expansion of powerful mobile devices define new issues for these software distributed systems. Thus, applications designed for mobile computing are expected to run in a highly heterogeneous and dynamic

environment, due the limited computing, storage and power capabilities of portable devices, the large variance in the communication bandwidth, and maybe the crucial factor, the mobility itself. In that sense, other mobile topics are emerging, e.g. the computing ubiquity, the natural interaction of the systems components and their intelligence. However, less attention has been paid to these last phenomena in the performance evaluation arena because the traditional software architectures for distributed applications are difficult to translate to ad-hoc communication environments [18]. Our position is that performance-related information must be considered not only for performance evaluation of the actors in a changing mobile environment, but also in scenarios where it is possible to reason about the performance activity in an intelligent ambient way and even take actions on it. Thus, the huge amount of knowledge that was researched under the software performance engineering may walk one step beyond to this cutting edge issue.

The remainder of the paper is organized as follows. In Section 2 we overview the different factors to be considered in the performance assessment of ambient intelligent applications (from now we name this approach PA-Ai). Section 3 of the paper summarizes related work, mainly giving an overview of work similar to the scope of this paper. The following section details the structure of the performance evaluation framework. Finally Section 5 summarizes the conclusions of this paper and provides an outlook to future work.

## 2. FACTORS TO CONSIDER IN PA-Ai

The following factors are the main issues to consider for performance assessment in ambient intelligence applications.

### 2.1 *Distributed intelligent applications*

A distributed application is an application which is executed based on a distributed system; therefore different parts of the application are processed on different machines. Usually the functionality of the architecture is mapped on the client-server paradigm. However, in mobile applications the client and server roles are not defined so specifically, some times devices are clients and some times they are servers. To add intelligence to such an application usually means that the system can learn from past experiences and make future decisions based onto this knowledge.

One possible scenario for a distributed intelligent application could be a *meeting coordination* system for office or congress use (MC scenario) [5]. In that scenario a congress participant enters the congress area. At that moment his personal digital assistants (PDA) automatically connects to the hotel server. It recognizes the conference participant, accesses his previous behavioral patterns, and immediately sends him information which could be useful for him. This information might be a room map when the conference is entered or the session agenda depending on the room being entered. It may contain a renewed session agenda which might have been altered due to short time changes. Additionally a list of the participants of the conference or a certain session can be offered, or supplementary information like presentation slides can be transferred to the attendees' mobile computer. The mobile device can also allow for communication with other congress members, for example with participants of the same session.

A second scenario might be useful for *office coordination* (OC scenario) [6]. A project manager can locate the members of his team using a "People Locating System (PLS)". This system is able to detect employees inside a companies building. When the project manager is scheduling a meeting the PLS is trying to locate all participants to be invited to be able to deliver them a message about the meeting schedule. Based on the participants behavior when receiving meeting information in the past according to their respective working circumstances, the PLS decides which type of message it delivers. When it finds two people together in a room with several others, it reasons that they are in a meeting and therefore decides to send them only a message notification to their PDA. Other members are located at their working place and thus are considered to be available; therefore they get the full text message onto their computers. Finally, two more members cannot be found on the company's site. The system hence accesses their appointment calendars and finds out that one of them has a meeting with a customer and thus should not be disturbed, and the other one is at his dentist. To both of them the system sends an email detailing the forthcoming meeting.

So the key difference between the traditional client-server architecture and these last scenarios is mainly how the information is represented in this changing environment. Whereas in traditional distributed software systems the representation is meant for computers to process information, i.e. syntactic level, in the ad hoc connected communication systems the representation allows devices to process and reason about information, i.e. semantic level. Therefore, it is necessary to get a semantic description of the components in the architecture [22].

Context-awareness systems not only consider the location but also any information that can be used to characterize the situation of the mobile devices, e.g. the system capabilities, the services offered and sought, the activities among devices and users, and their intentions.

### 2.2 *Mobile Devices*

Mobile devices, for example PDA's or Pocket PC's, are essential elements in future context-aware systems. Those devices are characterized by limited resources. They have

low processing power, constraints in memory capacity, communication bandwidth, and battery power. Hence, it is important to find a performance optimal architecture for applications using these limited devices.

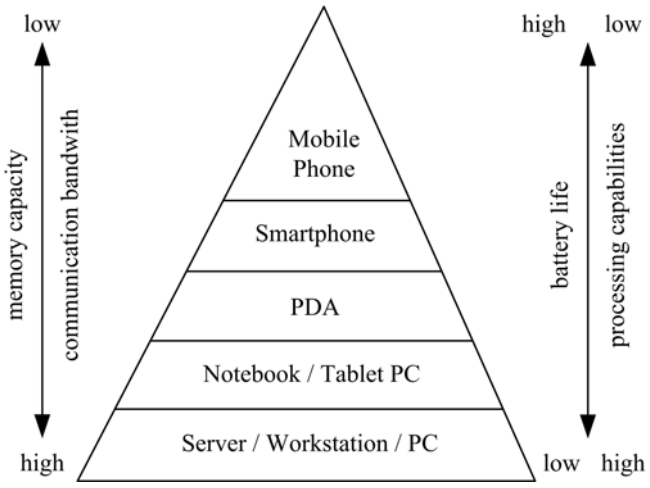


**Figure 1:** Device Capabilities

Concerning distributed intelligent applications we should at least consider five basic categories of devices which could be involved in performing various tasks for the application (see Figure 1). At the bottom of the pyramid the category consisting of immobile but powerful Servers, Workstations or PCs is located. The Notebook or Tablet PC on top of them is less powerful, but can be moved freely, only being limited by usually low battery endurance of a few hours. On the next higher layer PDAs provide less computing power, a limited user interface but stronger battery life up to usually about ten hours. Smartphones on the layer above have extended battery life, but even less processing capabilities and an even smaller user interface. On top of the pyramid are the mobile phones which can feature battery life of more than a week (not at heavy use), but offer only very limited processing power. Also the potential communication bandwidth and memory capacity is smallest on top of the pyramid and is increasing towards the base of it e.g. for the servers.

### 2.3 *Evaluation of architecture options*

Currently a number of well established software architectures are known, for example: (i) *Web-Services* are software components which are made useable via application servers. This model is also known as service-oriented architecture (SOA). (ii) In a *Client/Server architecture* resources are concentrated in one or a small number of nodes. So, in this model workload and bandwidth capabilities are unbalanced. (iii) In *Peer-to-Peer-Systems* workload and bandwidth demands will be distributed uniformly among the connected processors. (iv) *Component models* are based on building blocks which describe a well defined functionality. Such components can be accessed through interfaces (e.g. Corba, J2EE or .NET). (v) *Push-Systems* are used for efficient and timely distribution of information to a huge number of users. (vi) In *Event-Based-Systems* users are notified when determined events occur.

These architectures possess different characteristics like structure, degree of hierarchy or degree of coupling. When evaluating architecture options some of them will tend to be more adequate then others, but for one application there might be several suitable architecture options. So, given an application with its requirements and usage patterns a number of open questions arise. Is there only one adequate service architecture? How can several architecture options be assessed and qualified? Which design is the right one according to the given requirements and basic conditions? There may not exist a perfectly fitting architecture or a totally unsuitable one, but architectures which achieve a more or less suitable solution for a given problem and usage. A number of methods and techniques were developed for the evaluation of software architectures, for example: ATAM (Architecture Tradeoff Analysis Method, [10]), SAAM (Software Architecture Analysis Method, [11]), or ARID (Active Reviews for Intermediate Designs, [12]). However, for our purpose the major question is how to express the performance-related information in a context-awareness intelligent application.

## 3. RELATED WORK

UML diagrams that provide key information required for performance analysis are those that describe behaviour and resources together, therefore augmented sequence, activity, state chart and deployment annotated diagrams may express some performance information. A huge number of approaches have been proposed to derive performance models from software architecture specifications [2]. Basically, the concept can be used in an early stage of the software lifecycle. It uses the SPE architectural decision strategy. From annotated UML diagrams performance models are generated in the corresponding formalism (QN, SPN, SPA, etc.) and then they are offline evaluated through analytical, numerical or discrete-event simulation techniques. Following this procedure, [8] uses the SPE methodology for deriving performance models from software architecture specifications. In [7] a derivation of a QN model from SA is presented. This approach is based on Client/Server software performance evaluation (CLISSPE). In [13] an example to generate stochastic timed Petri net models from UML diagrams is shown. Finally [14] presents an example for the derivation of a performance model from an object-oriented design model. Due to the huge amount and the variety of proposals of 1.X UML performance extensions, new approaches are being developed for

performance modeling built from UML/SPT profile (Schedulability, Performance and Time) annotation [17].

Some performance analysis approaches have been reflected into mobile software architectures from annotated UML diagrams [2]. However, these solutions cover the mobility or location-awareness aspects, referring to the ability of the system to recognise the mobile components and the services (requested/offered) of the distributed system but not about the context or the ambient intelligence. Some performance tools and UML performance annotated design techniques have been connected through XML/XMI files [16].

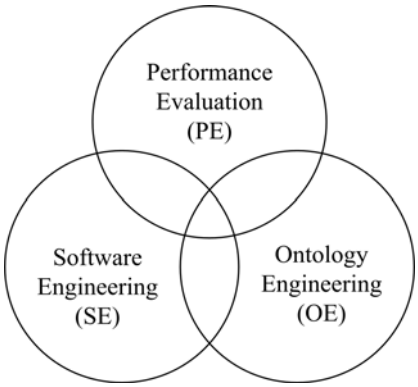


**Figure 2:** Intersection of PE, SE and OE areas

Several initiatives have been taken to deal with the topic of a joint terminology of context-awareness systems. Sponsored by the W3C, the web ontology language OWL seems to be a de facto standard. The OWL language builds on XML's ability to define customized tagging schemes and the flexible approach to representing data of RDF (Resource Description Framework). OWL is a language for defining and instantiating ontologies [23].

Figure 2 shows some of the research areas involved in the development of a framework to assess the performance of ambient intelligence applications. SE, PE and OE disciplines cannot provide a complete solution by themselves for certain topics, for example, the scope of SPE problems is located at the intersection between SE and PE. In this paper, we focus on the overlapping area between PE and OE and probably should be extended to all three disciplines.

Thus, critical issues in context-awareness research are context modeling, context intelligence (reasoning and knowledge) and context-privacy but other non-functional aspects are not considered, yet, e.g. context-aware performance assessment. In any case, software engineering has moved a bit since there are also early studies to map OWL into UML, but the approach on SPE may be different, as in next section we are going to overview.

# 4. STRUCTURE OF THE PERFORMANCE EVALUATION FRAMEWORK

There are several issues to be considered when defining a framework for the performance assessment of architectural choices in a context-awareness system (PA-Ai) that are similar to traditional SPE techniques: (i) It must be decided about the way the intelligent system is modeled and therefore, how to add the performance-related information (and which is interesting) into the specification with the minimal interference; (ii) Once the performance aspects of the system are depicted in the model, how to transform the architectural options onto a performance model and finally; (iii) how to evaluate the performance model of every choice. We are going to refer to this as Offline Performance Evaluation to distinguish if from the Online Adaptive Performance Brokerage.

## 4.1 *Off-line Performance Evaluation*

The framework shall provide an opportunity to compare different alternatives for architectures based on the capabilities of the involved devices and communication infrastructure. Thus an assessment of architecture options with respect to performance for various alternatives is done. This framework gives a strategy for a performance evaluation for architecture options based on relative performance predictions.

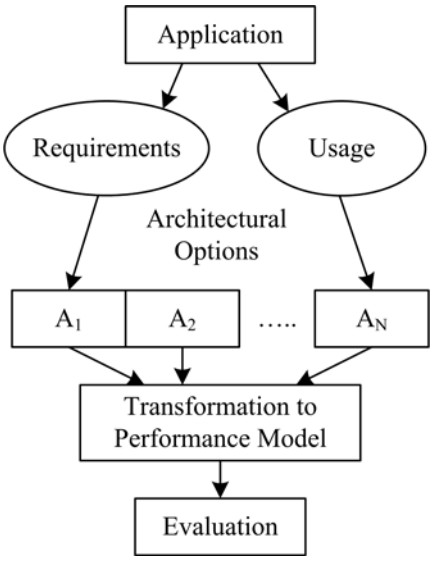


**Figure 3:** Framework Architecture

The overall architecture of the framework is depicted in Figure 3. The application determines the input parameters which are the requirements and perspective usage of the system. Depending on these parameters, several architectural options $A_i$ may be feasible. In the next stage,

these architectural options (based on an appropriate description) are transformed into a selected performance model which can be evaluated. This strategy does not differ from the traditional performance assessment for distributed applications although it has to consider the semantic representation of the information on the model.

An ontology is an explicit formal description of concepts in the domain composed of classes, properties of each class, and restrictions on properties. Therefore, it expresses the set of terms, entities, objects and classes and the relations between them with formal definitions. The use of ontologies contributes to knowledge sharing and reuse across systems. OWL ontologies are usually placed on web servers as web documents, which can be referenced by other ontologies and downloaded by applications that use these ontologies.

Our position is that performance-related information may be also declared through this new approach, not only for performance evaluation of the actors in a changing mobile environment, but also in scenarios where it is possible to reason about the performance activity in an intelligent ambient way and even take actions based on it.

On the other hand, ontologies can be used to build an information model, as some of the UML diagrams do, which allows the exploration of the information space in terms of the items which are represented, the associations between the items, the properties of the items, and even the links to documentation which describes and defines them (i.e., the external justification for the existence of the item in the model). That is to say that the ontology and taxonomy are not independent of the physical items they represent, but may be developed / explored in tandem. Thus, an ontology may consider performance-related information as description of the architecture of a system. Moreover, OWL should be compatible with other commonly used Web and industry standards. In particular, this includes XML and related standards (such as XML Schema and RDF), and possibly UML. Therefore we may exploit the interchange format between OWL and performance evaluation tools in the same manner as SPE engines. Figure 4 shows part of a simple example of OWL ontology encoded in RDF/XML.

```
<owl:Class rdf:ID="PDA">
  <rdfs:subClassOf rdf:resource="device" />
  ...
</owl:Class>

<owl:Class rdf:ID="performanceDescriptor" />

<owl:ObjectProperty rdf:ID="activity">
  <rdfs:domain rdf:resource="#device" />
  <rdfs:range
rdf:resource="#performanceDescriptor" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="locatedIn">
  ...
  <rdfs:domain
rdf:resource="http://www.w3.org/2002/07/owl#T
hing" />
  <rdfs:range rdf:resource="#building" />
</owl:ObjectProperty>

<owl:Class rdf:ID="demand" />
  <rdfs:subClassOf
rdf:resource="performanceDescriptor" />

<owl:DatatypeProperty rdf:ID="demandValue">
  <rdfs:domain rdf:resource="#demand" />
  <rdfs:range  rdf:resource="&xsd;float"/>
</owl:DatatypeProperty>

<demand rdf:ID="exp_average">
  <demandValue
rdf:datatype="&xsd;float">4.500</demandvalue>
</demand>
```

**Figure 4:** Simple OWL performance example

We provide a simple example of a vocabulary for performance-related information for the location information and average demand requirements of PDA devices (some information has been deleted due to space limitation of the text).

## 4.2 *On-line Performance Assessment*

However, proposing the use of OWL as a language to express similar performance annotated information as other de facto standards may not justify the effort. In this last case, only the syntactical view of OWL should be exploited.

One of the definitive features in ambient intelligence applications is the service discovery, i.e., functions offered by various mobile (e.g. mobile phones, PDAs, notebooks) and non-mobile devices (e.g. servers, printers, panels) that can be described and advertised, so that, they are sought-and-found by others. All of the current service discovery and capability description mechanisms (e.g. JINI, UPnP, JXTA, Bluetooth…) are based on ad-hoc representation schemes and rely heavily on standardization due to devices which were not necessarily designed to work together (such as ones built for different purposes, by different manufacturers, at a different time, etc.) as we experienced in the AKSIS project [5].

Being able to communicate at a high-level of abstraction with other devices, and reason about their services/functionality and performance is necessary for the complete evaluation of different architectural choices.

Thus, an ontology language will be used to describe the characteristics of devices, the means of access to such devices, the policy established by the owner for the use of a device, and other technical constraints and requirements that affect incorporating a device into a ubiquitous computing network. The needs established for DAML-S (DARPA Agent Markup Language) [25] and the RDF-based schemes

for representing information about device characteristics (namely, W3C's Composite Capability/Preference Profile (CC/PP) and WAP Forum's User Agent Profile (UAProf)) directly relate to this use case and the resource infrastructure which will support mobile applications and dynamically configure/negotiate ad-hoc networks. Thus, the performance information about resources, activities, actions, etc. in the context may be included as subproperties and datatypes in an extended vocabulary for OWL. This performance-related information and several simple operational rules and heuristic knowledge may be used for reasoning during execution about the performance of devices and services. Therefore, scenarios as OC or MC may be implemented through a team of context brokers. The context brokers would be running on stationary servers. A service discovery infrastructure will meet devices and servers, and the ontology will acquire information and reason about users, location, privacy and also performance. For example, in the OC scenario the ontology must include identifiable places in order to infer about location context. Reasoning about the spatial situation can predict performance improvements for example by mirroring services or automatically by disabling inactive device connections. To support reasoning with the device hardware/software descriptions, the ontology not only has to include profiles that would be extensions of [24] but also about PDAs and mobile phones to implement the MC scenario. Inferring about the device profiles may play an important role for capacity planning during context execution. The DAML ontology is a temporal ontology for expressing time-related properties. An extended OWL would have to consider this crucial information for performance prediction since it could be used to know the throughput of servers, the latency of a connection, the utilization of a device, etc. Moreover, location and temporal reasoning may be correlated for performance assessment purposes learning about inconsistencies among offered/required services in the scenarios.

## 5. CONCLUSION AND FUTURE WORK

This position paper tries to address the use of ontology as the solution to evaluate the performance of intelligent context-aware systems. Our preliminary study shows that OWL is not only a requirement for knowledge sharing in pervasive ambience, but also for acquiring performance-related information and the subsequent reasoning. However, the first step is to show that the syntactic use of ontologies for performance evaluation may incorporate the same information as annotated modeling languages in the SPE area. Thus, the off-line performance evaluation of architectural choices would be computed from the object properties and datatype definitions with performance constraints. The interconnection between the annotations and the performance tools for analytical solving or discrete-event simulation would use the XMI/XML interchange formats. Although this work is only overviewed in this paper, it could represent a primary step for evaluating the performance of context-awareness systems.

A more ambitious project would be the utilization of context brokers in order to assess performance during context execution. The advantage of the OWL description of the ambient may use the semantics to infer performance knowledge. Even the off-line performance evaluation relies on the annotated constraint values; it seems to be possible to get information on-line about the relationships in the context and to reason about them. Thus, a team of context brokers would implement the architecture in various aspects of pervasive computing, e.g. location, timing, device profiling, etc. and performance.

## 6. REFERENCES


[1] Bass L., Clements P., and Kazman R., *Software Architecture in Practice*, Addison-Wesley, 1998

[2] Cortellessa V., and Mirandola R., "PRIMA-UML: A Performance Validation Incremental Methodology on early UML Diagrams", *Science of Computer Programming*, vol. 44, pp. 101-129, 2002

[3] Dustdar S., Gall H., and Hauswirth M., *Software-Architekturen für Verteilte Systeme*, Springer-Verlag, 2003

[4] Gomaa, H., *Software Design Methods for Concurrent and Real-time Systems*, The SEI Series in Software Engineering, N. Habermann (ed.), Addison-Wesley, Reading, Massachusetts, 1993

[5] Hummel K.A., "Meeting Coordination", http://www.ani.univie.ac.at/~karin/ambience/scenarios/meeting.pdf

[6] Hummel K.A., "Office Communication", http://www.ani.univie.ac.at/~karin/ambience/scenarios/communication.pdf

[7] Menascè D.A., and Gomaa H., "On a Language Based Method for Software Performane Engineering of Client/Server Systems", in Proceedings of the 1st International Workshop on Software and Performance, pp. 63-69, 1998

[8] Smith C.U., and Williams L.G., "Performance Evaluation of a Distributed Software Architecture", in Proceedings of the 1st International Workshop on Software and Performance, pp. 164-177, 1998

[9] Smith C.U., and Williams L.G., *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*, Addison-Wesley, 2002

[10] Kazman R., Klein M. Barbacci M., and Lipson H. Longstaff T. Carriere S.J., "The Architecture Tradeoff Analysis Method", in Proceedings of the 4st IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), pp. 68-78, 1998

[11] De Simone M., and Kazman R., "Software Architecture Analysis: An Experience Report", in Proceedings of the

1995 conference of the Centre for Advanced Studies on Collaborative research, 1995
[12] Clements P., "Active Reviews for Intermediate Designs", Technical Note CMU/SEI-2000-TN-009, Software Engineering Institute, Carnegie Mellon University
[13] King P., and Pooly R., "Derivation of Petri Net Performance Models from UML Specification of Communication Software", in Proceedings of the 1997 Computer Measurement Group Conference, 1997
[14] Smith C.U, and Williams L.G., "Performance Engineering Evaluation of Object Oriented Systems with SPE•ED", in *Computer Performance Evaluation: Modelling Techniques and Tools*, Springer-Verlag, 1997
[15] Object Management Group (OMG): Unified Modeling Language Specification, version 1.3, http://www.omg.org/uml
[16] Object Management Group (OMG): XML Metadata Interchange (XMI) Specification, version 1.2, http://www.omg.org/cgi-bin/
[17] Object Management Group (OMG): UML Profile for Schedulability, Performance and Time Specification, March 2002
[18] Beer, W., Christian, V., Ferscha, A. and Mehrmann L., "Modeling Context-Aware Behavior by Interpreted ECA rules". In Proceedings of Euro-Par 2003, H. Kosch, L. Böszörményi and H. Hellwagner (Eds.), LNCS 2790, pp. 1064-1073, 2003
[19] Proceedings of the Second Workshop on Software and Performance (WOSP 2000), Ottawa, Canada, September 2000, ACM Press
[20] Proceedings of the Third Workshop on Software and Performance (WOSP 2002), Rome, Italy, July 2002, ACM Press
[21] Proceedings of the Fourth Workshop on Software and Performance (WOSP 2004), San Francisco, USA, January 2004, ACM Press
[22] Berners-Lee, T., Hendler, J. and Lassila, O. "The Semantic Web", Scientific American, May 2001
[23] Van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel- Schneider, P.F. and Stein, L.A. "OWL Web Ontology Language reference", 2002, http://www.w3c.org/TR/owl-ref/
[24] Foundation for Intelligent Physical Agent. FIPA Device Ontology Specification, pc00091a edition, 2001
[25] Hobbs, J.R. "A DAML Ontology of Time", http://www.cs.rochester.edu/daml